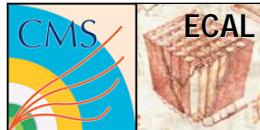


LCG Conditions DB API

Ricky Egeland, University of Minnesota
USCMS-DB Meeting

November 12, 2004



Outline



- LCG CondDB Current API vs. Planned Changes
 - See Andrea Valassi's Oct 13 Talk
<http://agenda.cern.ch/fullAgenda.php?ida=a044296>
- Installation Issues
- Performance Issues
- What's Next



Schema review - 2 (IMO)



<table border="1"><tr><td>folderID</td><td>since</td><td>till</td><td>(tag)</td><td>dataID</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	folderID	since	till	(tag)	dataID																				<table border="1"><tr><td>dataID</td><td>BLOB</td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>	dataID	BLOB							<p>1- BLOB API Versioning. All folders in one IOV table. External payload table.</p>	CondDBOracle				
folderID	since	till	(tag)	dataID																																			
dataID	BLOB																																						
<table border="1"><tr><td>since</td><td>till</td><td>(tag)</td><td>dataID</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	since	till	(tag)	dataID																	<table border="1"><tr><td>dataID</td><td>BLOB</td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>	dataID	BLOB							<p>1- BLOB API Versioning. One IOV table per folder. External payload table (BLOBS).</p>	CondDBMySQL								
since	till	(tag)	dataID																																				
dataID	BLOB																																						
<table border="1"><tr><td>since</td><td>till</td><td>(tag)</td><td>dataID</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	since	till	(tag)	dataID																	<table border="1"><tr><td>dataID</td><td>user1</td><td>user2</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	dataID	user1	user2										<p>2- Extended API (STRUCTTAG) Versioning. No ChannelID. One IOV table per folder. External payload table (user fields).</p>					
since	till	(tag)	dataID																																				
dataID	user1	user2																																					
<table border="1"><tr><td>channelID</td><td>since</td><td>till</td><td>user1</td><td>user2</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	channelID	since	till	user1	user2																				<p>3- Extended API (STRUCTID) No versioning. ChannelID. One IOV table per folder. External payload table (user fields).</p>														
channelID	since	till	user1	user2																																			
<table border="1"><tr><td>channelID</td><td>since</td><td>till</td><td>(tag)</td><td>user1</td><td>user2</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	channelID	since	till	(tag)	user1	user2																			<p>New proposed API One IOV table per folder. Versioning (unless disabled). Channel ID (unless disabled). External payload (unless inlined) - internal implementation detail</p>														
channelID	since	till	(tag)	user1	user2																																		
<table border="1"><tr><td>channelID</td><td>since</td><td>till</td><td>(tag)</td><td>dataID</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	channelID	since	till	(tag)	dataID																				<table border="1"><tr><td>dataID</td><td>user1</td><td>user2</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	dataID	user1	user2										<p>Seen from outside, user interface offers a single consistent view.</p>	
channelID	since	till	(tag)	dataID																																			
dataID	user1	user2																																					



API review - 1 (IMO)



```
class ICondDBObject {  
    CondDBKey validSince() const;  
    CondDBKey validTill() const;  
    ...  
    void data( string blob ) const;  
}
```

= Original (BLOB) API

= Extended (ICondDBTable) API

```
class ICondDBTable {  
    enum cdb_types {cdbNull=0, cdbBool, cdbInt, ...};  
    int getNames (vector<string>& names);  
    int getTypes (vector<cdb_types>& types);  
    int setName (unsigned n_column, string& name);  
    int setType (unsigned n_column, cdb_types type);  
    ...  
    int getRow (unsigned rowNumber, vector<string>& values);  
    int [get/set]Cell (unsigned n_column, unsigned n_row, [int/float/..] data);  
    ...  
    int get[Since/Till]Time (unsigned n_row, SimpleTime& time);  
    int set[Since/Till]Time (vector<SimpleTime>& times);  
}
```

1. Keep one interface ([ICondDBObject](#)) for BLOBS and user fields: use [AttributeList](#)
 - Replace `data()` by method returning an [AttributeList](#) (intrinsic payload)
 - Support all simple C++ types (`int`, `float`, `string`...) and only them: no user-defined types
 - BLOB type presently not in [AttributeList](#), should be included to support CondDB BLOBS (LHCb)
 - IMO: remove POOL Token from [AttributeList](#), store as strings, interpret in external component
 - Maximise reuse of existing LCG solutions (no dependency on POOL if moved to SEAL)
2. Differentiate schema vs. data: use [AttributeListSpecification](#)
 - Encapsulated in an [ICondDBObjectSpecification](#)?
3. Differentiate one vs many: a table is just a set of objects



API review - 2 (IMO)



Create a folder

```
createCondDBFolder(  
    string fullPath,  
    string attributes = "",  
    string description = "",  
    bool parents = false);
```

```
createCondDBFolder(  
    string fullPath,  
    ICondDBTable* table,  
    string attributes = "",  
    string description = "",  
    bool parents = false,  
    folder_types filetype = STRUCT);
```

Store an object

```
storeCondDBObject(  
    string folder,  
    ICondDBObject* CondObject );
```

```
storeCondDBObject(  
    string folder,  
    ICondDBTable* table);
```

= Original (BLOB) API

= Extended (ICondDBTable) API

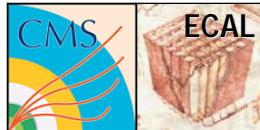
Find an object

```
findCondDBObject(  
    ICondDBObject*& oblock,  
    string folder,  
    CondDBKey& point,  
    string tag = "");
```

```
findCondDBObject(  
    ICondDBTable *table,  
    string folder,  
    CondDBKey& point,  
    string id = "",  
    string& selection = "",  
    vector<string>* nullValues = 0,  
    string tag = "");
```



- 1- Keep one interface (ICondDBObject) for BLOBs and user fields: use AttributeList
- 2- Differentiate schema vs. data: use ~ AttributeListSpecification at creation time
- 3- Differentiate one vs. many: get ~ ICondDBObjectSet (or vector<ICondDBObject>) when reading; storing a "table" is just a sequential insertion of many rows?
- 4- IF (?) selection on data content is needed, do not mix it with lookup by time and tag
OK to insert concept of ID and extra folder options (inline/external, online/versioning)



API Code - Today



```
// make database
ICondDBMgr* condDBmgr = CondDBOracleMgrFactory::createCondDBMgr();
condDBmgr->init( "user=AAA,passwd=BBB,db=CCC,cond_db=DDD" );

// make folder
ICondDBBasicFolderMgr* condFolderMgr = condDBmgr->getCondDBBasicFolderMgr();
condFolderMgr->createCondDBFolder( "/folder/structure" );

// make object
ICondDBObject* condObject =
    CondDBOracleObjectFactory::createCondDBObject(3, 10,
                                                "dataGoesHere",
                                                "sample description");

// write object
ICondDBBasicDataAccess* condDataAccess = condDBmgr->getCondDBBasicDataAccess();
condDataAccess->storeCondDBObject( "/folder/structure", condObject );

// read object
condDataAccess->findCondDBObject( condObject, "/folder/structure", 7);

// use object
CondDBKey since( oCondObject->validSince() );
CondDBKey till( oCondObject->validTill() );
string data = "";
condObject->data(data);
```



API Code - Tomorrow



Very early example - not yet implemented!

```
// make/connect database

cool::TechnologyID techId ( "RAL" );
boost::shared_ptr<IDatabaseMgr> mgr =
    cool::DatabaseMgrFactory::createMgr( techId );

cool::DatabaseID dbId
( "mysql://tcphost;user=AAA;password=BBB" );
boost::shared_ptr<cool::IDatabase> db =
    mgr->createDatabase( dbId, dbPhysicalAttributes ); }
```



API Code - Tomorrow (2)



Very early example - not yet implemented!

```
// create folder and define type  
  
cool::FolderID folderId( "/myfolder" );  
  
pool::AttributeListSpecification payloadSpec;  
payloadSpec.push_back( "I","int" );  
payloadSpec.push_back( "S","string" );  
payloadSpec.push_back( "X","float" );  
  
boost::shared_ptr<cool::IFolder> folder =  
    db->createFolder( folderId, payloadSpec );
```



API Code - Tomorrow (3)



Very early example - not yet implemented!

```
// make data object and write

pool::AttributeList payload0( payloadSpec );
    int i0 (1);
    string s0 ("ABCDEF");
    float x (11.);
    payload0[ "I" ].setValue(i0);
    payload0[ "S" ].setValue(s0);
    payload0[ "X" ].setValue(x0);

    ValidityKey since( 5 );
    ValidityKey till( 7 );
    folder->storeObject( since, till, payload0 );
```



API Code - Tomorrow (4)



Very early example - not yet implemented!

```
// find object, read data

boost::shared_ptr<cool::Object> object =
    folder->find( 6 );
pool::AttributeList payload = object->payload();

int i;
string s;
float x;
payload[ "I" ].getvalue(i);
payload[ "S" ].getvalue(s);
payload[ "X" ].getvalue(x);
```

pros

- More flexible data definition
- Choice of DB backend
- In development = our input counts

cons

- More packages
- Not ready yet

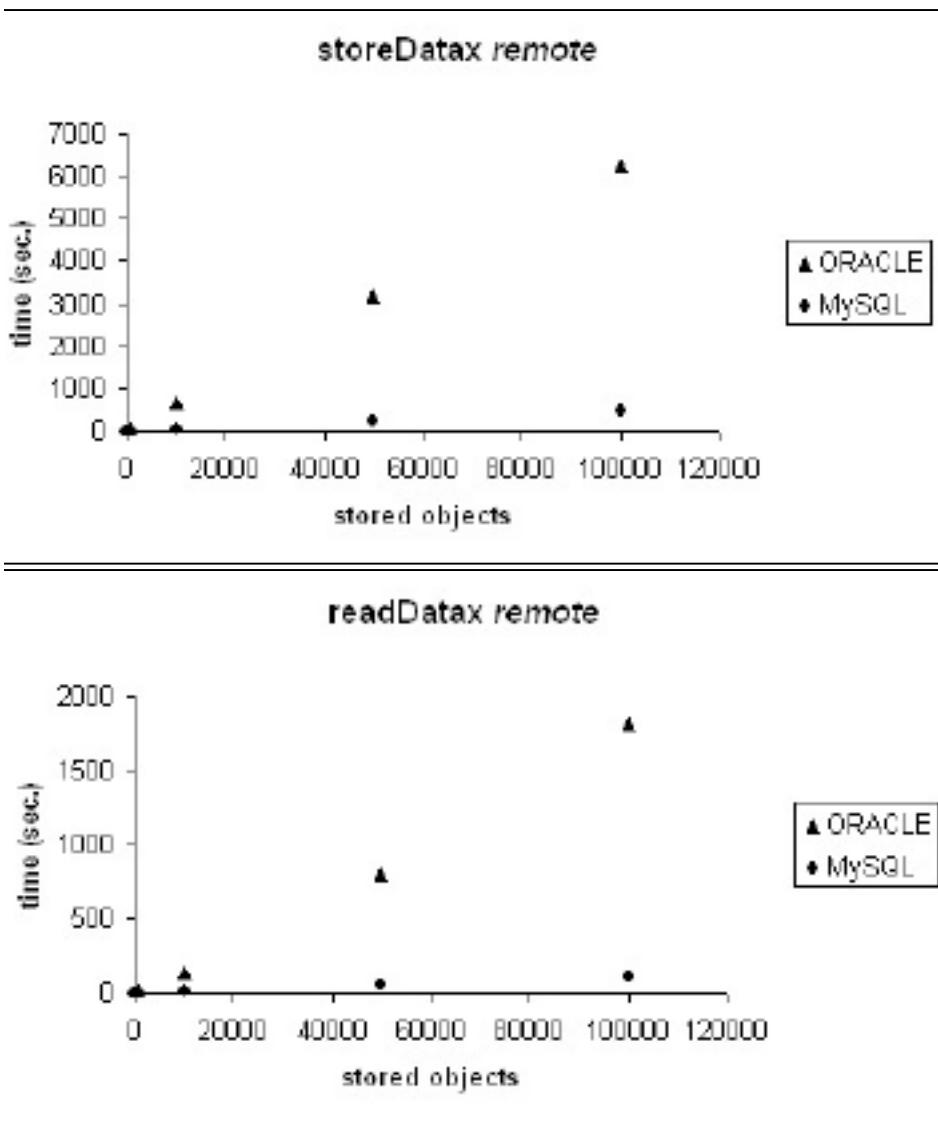


Installation Issues



- Current (and future) Conditions DB uses SCRAM for installation
 - Provided configurations heavily tied to CERN systems (architecture, AFS)
- I have installed CondDB on my local machine (SLC3)
 - Created tools-STANDALONE.conf file
 - HOWTO given to Andrea, should go up on web
- So far no luck compiling on Windows (Gennadiy), where our DCS programs run

Performance Issues



Tests done by ATLAS Lisbon Group

<http://kdataserv.fis.fc.ul.pt/ATLAS/>

Server Hardware

P4 1.6 GHz

756 MHz Ram 133 MHz

30 GB HDD ATA100 7200 RPM

Server Software

Suse 8.0, linus 2.4.18

Oracle 9.2.0.1.0

MySQL 3.23.48



Performance Issues (2)

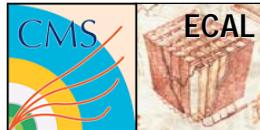
Intensive Usage		<i>Oracle (remote)</i>	<i>MySQL (remote)</i>
<i>createFolderx</i>		real 0m11.857s	real 0m0.072s
<i>storeDatax</i>	<i>10 objs</i>	real 0m5.434s	real 0m1.127s
	<i>100 objs</i>	real 0m15.820s	real 0m2.345s
	<i>1.000 objs</i>	real 1m2.850s	real 0m6.341s
	<i>10.000 objs</i>	real 11m12.554s	real 0m56.929s
	<i>50.000 objs</i>	real 52m49.003s	real 3m53.565s
	<i>100.000 objs</i>	real 104m13.283s	real 8m16.510s
<i>readDatax</i>	<i>10 objs</i>	real 0m0.311s	real 0m0.082s
	<i>100 objs</i>	real 0m1.531s	real 0m0.146s
	<i>1.000 objs</i>	real 0m13.571s	real 0m1.084s
	<i>10.000 objs</i>	real 2m19.243s	real 0m11.021s
	<i>50.000 objs</i>	real 13m12.387s	real 0m49.572s
	<i>100.000 objs</i>	real 30m14.354s	real 1m45.335s

E
C
A
L

Type	N_barrel	1/Freq [s]	Mbytes/day
High Voltage	2448	10	***103.4
Low Voltage	29376	120	***103.4

Tests done by ATLAS Lisbon Group
<http://kdataserv.fis.fc.ul.pt/ATLAS/>

*** reduce by 1/50 ?



Performance Issues (3)



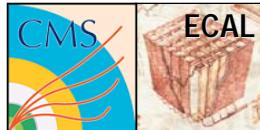
“Keep one main focus for the project - offline analysis. Online option too, but not the main performance target”

- A. Valassi, LCG, Oct 13

“Most of the conditions data stays at the experiment and is NOT used for offline reconstruction”

- F. Glege, CMS, Nov 3

- How will conditions data be used?
- Unless performance improves or data reduces, online will not be an option for ECAL



What's Next (ECAL)



- Received draft list of database needs from ECAL DCS
- In contact with ECAL Laser Monitoring
 - Needs to be determined what is kept in Conditions DB
- Need to make well-defined data inventories and schema (folder structure?)
- Meeting at CERN on Tuesday Nov 16 will hopefully better **define** ECALs database needs and goals
 - <http://agenda.cern.ch/fullAgenda.php?ida=a044973>
- **When will CMS converge on a Conditions DB system?**